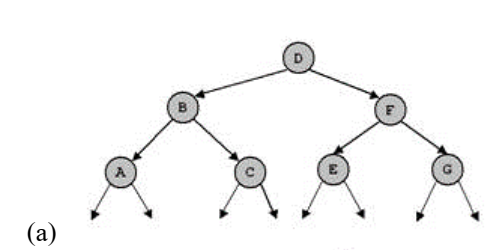
Họ và tên: Đỗ Thế Hiếu

MSSV: SE170041

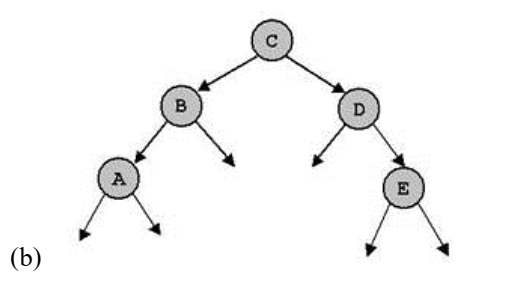
LAB 4

**Writing exercise:**

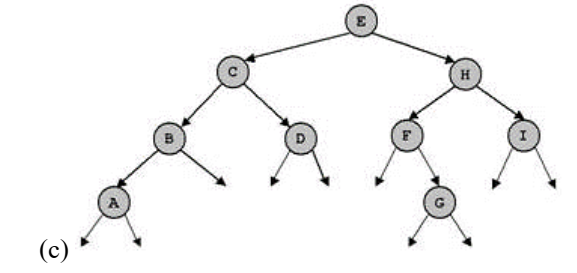
**1) Give the preorder, inorder, postorder, and level-order traversals of the following binary trees.**



* **Preorder: D, B, A, C, F, E, G**
* **Inorder: A, B, C, D, E, F, G**
* **Postorder: A, C, B, E, G, F, D**
* **Level-order: D, B, F, A, C, E, G**



* **Preorder: C, B, A, D, E**
* **Inorder: A, B, C, D, E**
* **Postorder: A, B, E, D, C**
* **Level-order: C, B, D, A, E**



* **Preorder: E, C, B, A, D, H, F, G, I**
* **Inorder: A, B, C, D, E, G, F, H, I**
* **Postorder: A, B, D, C, G, F, I, H, E**
* **Level-order: E, C, H, B, D, F, I, A, G**

**2) For each of the following key sequences create the binary search tree obtained when the keys are inserted one-by-one in the order given into an initially empty tree:  
(1) 4, 3, 1, 11, 5, 9, 2, 6, 15, 12.  
(2) 12, 7, 1, 3, 2, 5, 10, 8, 6, 9.  
Give the preorder, inorder, postorder,and level-order traversals of the created binary trees.  
In (1) delete keys 2, 3 and 11. In (2) delete keys 5, 6 and 7. After each time of deleting, give the above traversals**

**(1) 4, 3, 1,11, 5, 9, 2, 6, 15, 12.**



* Preorder: 4, 3, 1, 2, 11, 5, 9, 6, 15, 12
* In-order: 1, 2, 3, 4, 5, 6, 9, 11, 12, 15
* Post-order: 2, 1, 3, 6, 9, 5, 12,15 ,11 ,4
* Level-order: 4 ,3 ,11 ,1 ,5 ,15 ,2 ,9 ,12 ,6

**In (1) delete keys 2, 3 and 11**

* **4, 1, 5, 9, 6, 15, 12.**



* Preorder: 4,1,12,5,9,6,15
* In-order: 1,4,6,9,5,12,15
* Post-order: 1,6,9,5,15,12,4
* Level-order: 4,1,12,5,15,9,6

**(2) 12, 7, 1, 3,2, 5, 10, 8, 6, 9.**



* Preorder: 12, 7, 1, 3, 2, 5, 6, 10, 8, 9
* In-order: 1, 2, 3, 5, 6, 7, 8, 9, 10, 12
* Post-order: 2, 6, 5, 3, 1, 9, 8,10 ,7 ,12
* Level-order: 12 ,7 ,1 ,10 ,3 ,8 ,2 ,5 ,9 ,6

**In (2) delete keys 5, 6 and 7.**

* **12, 1, 3,2,10, 8, 9.**



* Preorder: 12,8,1,3,2,10,9
* In-order: 1,2,3,8,9,10,12
* Post-order: 2,3,1,9,10,8,12
* Level-order: 12,8,1,10,3,9,2

**3) For each of the key sequences in question 2 create the AVL tree obtained when the keys are inserted one-by-one in the order given into an initially empty tree.  
Give the preorder, inorder, postorder, and level-order traversals of the created  
binary trees.**

**(1) 4, 3, 1,11, 5, 9, 2, 6, 15, 12.**



* Preorder: 5,3.1.2.4.9.6.12.11.15
* In-order: 2,1,3,4,5,6,9,11,12,15
* Post-order: 2,1,4,3,6,11,15,12,9,5
* Level-order: 5,3,9,1,4,6,12,2,11,15

**In (1) delete keys 2, 3 and 11**

* **4, 1, 5, 9, 6, 15, 12.**



* Preorder: 5,4,1,9,6,12,15
* In-order: 1,4,5,6,9,12,15
* Post-order: 1,4,6,15,12,9,5
* Level-order: 5,4,9,1,6,12,15

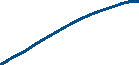
**(2) 12, 7, 1, 3,2, 5, 10, 8, 6, 9.**



* Preorder: 7,3,2,1,5,6,10,8,9,12
* In-order: 1,2,3,5,6,7,9,8,10,12
* Post-order: 1,2,6,5,3,9,8,12,10,7
* Level-order: 7,3,10,2,5,8,12,1,6,9

**In (2) delete keys 5, 6 and 7.**

* **12, 1, 3,2,10, 8, 9.**



* Preorder: 8,2,1,3,10,9,12
* In-order: 1,2,3,8,9,10,12
* Post-order: 1,3,2,9,12,10,8
* Level-order: 8,2,10,1,3,9,12

**4) The tree (c) in question 1 is height balanced (AVL). Delete the node D and perfom  
necessary operations so that the tree is still AVL after deletion.**

A picture containing sketch, drawing, circle, white

Description automatically generated



**5) Consider a nearly complete binary tree with n nodes.  
a. For what values of n is a nearly complete binary tree a full binary tree?**

**- Answer: value of n: odd number  
b. For what values of n is a nearly complete binary a complete (perfect) binary tree?**

**- Answer: 2^(h+1) – 1 – n <= 2^h (where h is the height of the tree)**

**6) Suppose we have numbers between 1 and 100 in a binary search tree and we  
want to search for the number 57 in the tree. Could the following sequence be the  
sequence of nodes to be examined?  
2, 90, 63, 70, 68, 72, 57**

**- Answer:**

**It cannot be a binary search sequence to find node 57 in a binary tree from 1 to 100. Let's parse this string:**

**+ Take root as 2, because 57 > 2, the algorithm moves to the right**

**+ Next, find and consider at node 90, 57 < 90, so the algorithm moves left**

**+ Next, find and consider at node 63, 57 < 63, so the algorithm moves to the left**

**+ Next, 70 can't be found on the left side of 63**

**-> Could not find node 57 in this sequence: 2, 90, 63, 70, 68, 72, 57**